

# CI/CD Deployment for Springboot Application

Prototype of the Application

Name : Anurag Sharma

GitHub : <https://github.com/Instantgaming2356/JAVAFSD-Project05>

This documentation will guide you through the steps involved in setting up a CI/CD pipeline for a Spring Boot application using Jenkins as the CI/CD tool and AWS EC2 instance as the hosting server. The application code will be fetched from a GitHub repository, and the deployment will be automated using Jenkins.

## Sprint Planning

The Implementation is done in Eight sprints which are mentioned below

Prerequisites:

A GitHub account to store the source code.

A Jenkins server with admin access.

An AWS account to create an EC2 instance.

Basic knowledge of Spring Boot, AWS EC2, and Jenkins.

Sprint involved:

### **Sprint 1: Clone the GitHub repository:**

Clone the repository containing the source code of the Spring Boot application to the local system. This will be used by Jenkins to build the application and create a deployment package.

### **Sprint 2: Install necessary plugins:**

Install the following plugins in the Jenkins server:

Maven Integration Plugin

Git plugin

These plugins will be used to build the application and fetch the source code from the GitHub repository.

### **Sprint 3: Create a Jenkins job:**

Create a new Jenkins job by navigating to Jenkins Dashboard > New Item > Freestyle

project.

In the Source Code Management section, select Git and provide the URL of the GitHub repository where the code is stored.

In the Build section, select Invoke top-level Maven targets and provide the necessary Maven goals to build the application. For example, clean install.

#### **Sprint 4: Configure AWS EC2 instance:**

Create an EC2 instance in the AWS account and configure the security group to allow HTTP traffic on port 8080.

Install Java and Tomcat on the EC2 instance to run the Spring Boot application.

#### **Sprint 5: Install AWS CLI:**

Install the AWS CLI on the Jenkins server to automate the deployment process.

#### **Sprint 6: Configure AWS credentials:**

Add the AWS access key and secret key to the Jenkins server using the AWS CLI.

#### **Sprint 7: Create a deployment job:**

Create a new Jenkins job to deploy the Spring Boot application to the EC2 instance. In this job, configure the following steps:

Fetch the latest deployment package from the Jenkins workspace.

Copy the deployment package to the EC2 instance using the AWS CLI.

Start the Tomcat server on the EC2 instance to run the application.

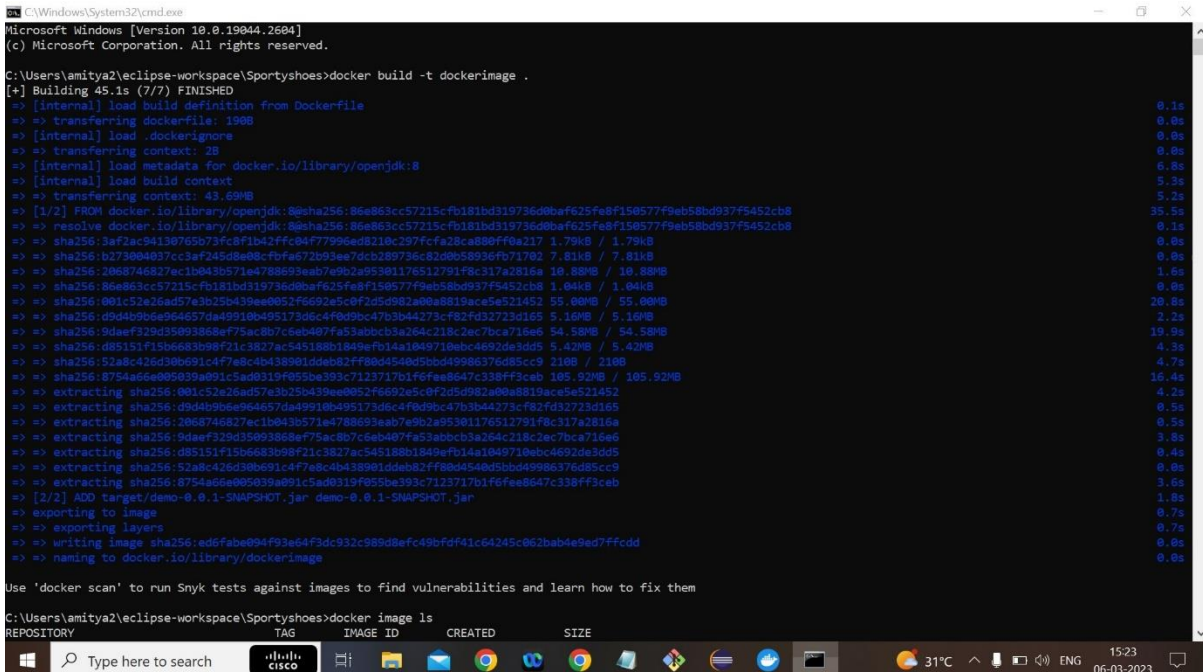
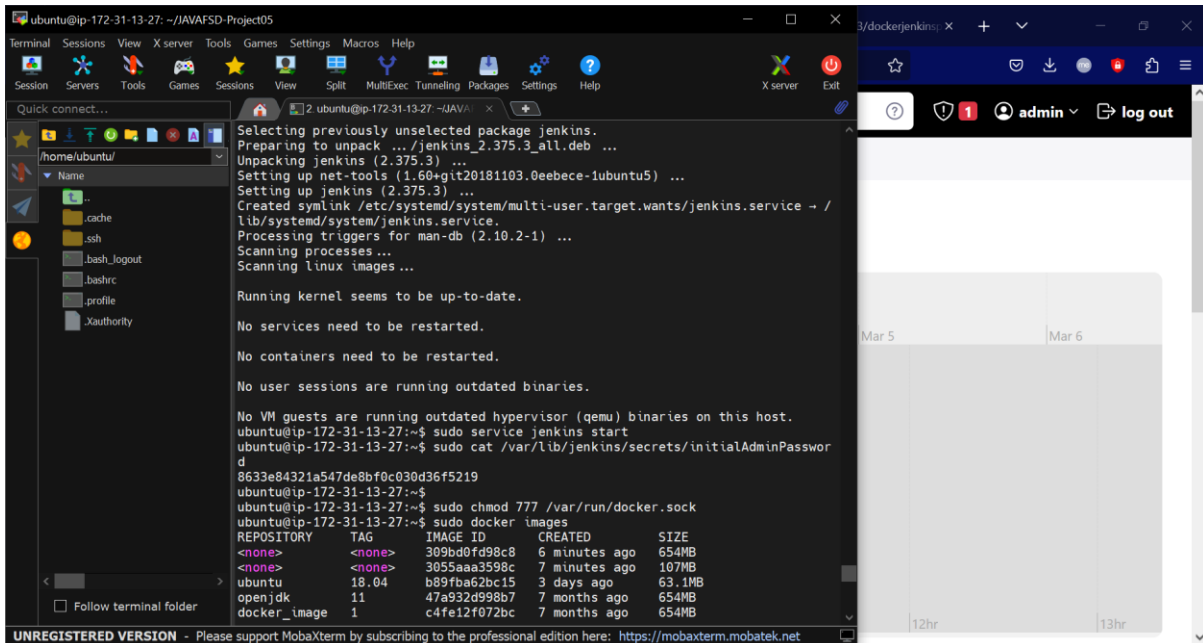
#### **Sprint 8: Test the deployment:**

Run the deployment job and check if the application is deployed successfully on the EC2 instance. Access the application using the public IP address of the EC2 instance.

#### **Conclusion:**

By following the above steps, you can automate the integration and deployment of a Spring Boot application using Jenkins and AWS EC2 instance. This will reduce the manual effort required for deployment and increase the efficiency of the development process.

# Screenshot



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\amitya2\workspace\Sportyshoes>docker build -t dockerimage .
[+] Building 45.1s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 190B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/openjdk:8
=> [internal] load build context
=> => transferring context: 43.69MB
=> [1/2] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0bf625fa8f150577f9eb58bd937f5452cb8
=> resolve docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0bf625fa8f150577f9eb58bd937f5452cb8
=> sha256:3af2ac9413675b73fc8f1b42ffc04777996ad8218c297fca28ca808f0a217 1.79kB / 1.79kB
=> sha256:b273804837cc3af245d8a08cf0fa672b93ee7dcb289736c82d0b58936fb71702 7.81kB / 7.81kB
=> sha256:2968746827ec1b043b571e4788683eab7e9b2a95301176512791f8c317a2816a 10.88MB / 10.88MB
=> sha256:86e863cc57215cfb181bd319736d0bf625fa8f150577f9eb58bd937f5452cb8 1.04kB / 1.04kB
=> sha256:001c52e26ad57e3b25b439ee0052f6692a5c0f2d5d982a00e8619ace5e521452 55.00MB / 55.00MB
=> sha256:d9d4b9b6e964657da49910b495173d6c4f0d9bc47b3b44273cf82fd32723d165 5.16MB / 5.16MB
=> sha256:9dae732d35993868e75ac8b7c8eb407fa53abbc3a264c218c2ec7bca716ae 54.58MB / 54.58MB
=> sha256:d85151f15b6683b98f21c3827ac545188b1849efb14a1049710ebc4692de3dd5 5.42MB / 5.42MB
=> sha256:52a8c426d30b691c4f7e8c4b438901ddeb82ff8bd4540d5b0d49986376d85cc9 210B / 210B
=> sha256:8754a66e005039a091c5ad0319f055be393c7123717b1f6fee8647c338ff3ceb 105.92MB / 105.92MB
=> extracting sha256:001c52e26ad57e3b25b439ee0052f6692a5c0f2d5d982a00e8619ace5e521452
=> extracting sha256:d9d4b9b6e964657da49910b495173d6c4f0d9bc47b3b44273cf82fd32723d165
=> extracting sha256:2968746827ec1b043b571e4788683eab7e9b2a95301176512791f8c317a2816a
=> extracting sha256:9dae732d35993868e75ac8b7c8eb407fa53abbc3a264c218c2ec7bca716ae
=> extracting sha256:d85151f15b6683b98f21c3827ac545188b1849efb14a1049710ebc4692de3dd5
=> extracting sha256:52a8c426d30b691c4f7e8c4b438901ddeb82ff8bd4540d5b0d49986376d85cc9
=> extracting sha256:8754a66e005039a091c5ad0319f055be393c7123717b1f6fee8647c338ff3ceb
[2/2] ADD target/demo-0.0.1-SNAPSHOT.jar demo-0.0.1-SNAPSHOT.jar
=> exporting to image
=> exporting layers
=> writing image sha256:ed6fabe094f93e64f3dc932c989d8efc48bdfd41c64245c862bab4e9d7ffcd
=> naming to docker.io/library/dockerimage

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\amitya2\workspace\Sportyshoes>docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
dockerimage	latest	ed6fabe094f9	31 seconds ago	570MB
mysql	latest	4f06b49211c0	10 days ago	530MB
dev_utils-celery_app	latest	cdbabfb8bbec	2 weeks ago	2.45GB
<none>	<none>	3d281705af01	5 weeks ago	2.45GB
<none>	<none>	67b60bdabf65	5 weeks ago	2.45GB
<none>	<none>	850b262a58cf	2 months ago	2.48GB
<none>	<none>	a23e1ad8d671	2 months ago	2.45GB
<none>	<none>	32654e4bf7e9	2 months ago	2.45GB
<none>	<none>	90cc2271851e	2 months ago	2.45GB
<none>	<none>	c04cee863c1	2 months ago	2.45GB
<none>	<none>	c7840d5066fd	2 months ago	2.45GB
<none>	<none>	78cc35e0b4a8	2 months ago	2.45GB
<none>	<none>	36d5662ea5cb	2 months ago	2.45GB
dev_utils-app	latest	775ad237db4e	2 months ago	2.43GB
<none>	<none>	44f05b07755f	2 months ago	2.43GB
<none>	<none>	3e11daa8ebb9	2 months ago	317MB
dev_utils-oneuxdb-dev	latest	c07d710b089d	2 months ago	317MB
hello-world	latest	fb5d9f0a6a5	17 months ago	13.3kB
containers.cisco.com/ifs_omega/redis	latest	de974760db2	23 months ago	105MB

```
C:\Windows\System32\cmd.exe
C:\Users\amitya2\workspace\Sportyshoes>docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
dockerimage latest ed6fabe094f9 31 seconds ago 570MB
mysql latest 4f06b49211c0 10 days ago 530MB
dev_utils-celery_app latest cdbabfb8bbec 2 weeks ago 2.45GB
<none> <none> 3d281705af01 5 weeks ago 2.45GB
<none> <none> 67b60bdabf65 5 weeks ago 2.45GB
<none> <none> 850b262a58cf 2 months ago 2.48GB
<none> <none> a23e1ad8d671 2 months ago 2.45GB
<none> <none> 32654e4bf7e9 2 months ago 2.45GB
<none> <none> 90cc2271851e 2 months ago 2.45GB
<none> <none> c04cee863c1 2 months ago 2.45GB
<none> <none> c7840d5066fd 2 months ago 2.45GB
<none> <none> 78cc35e0b4a8 2 months ago 2.45GB
<none> <none> 36d5662ea5cb 2 months ago 2.45GB
dev_utils-app latest 775ad237db4e 2 months ago 2.43GB
<none> <none> 44f05b07755f 2 months ago 2.43GB
<none> <none> 3e11daa8ebb9 2 months ago 317MB
dev_utils-oneuxdb-dev latest c07d710b089d 2 months ago 317MB
hello-world latest fb5d9f0a6a5 17 months ago 13.3kB
containers.cisco.com/ifs_omega/redis latest de974760db2 23 months ago 105MB

C:\Users\amitya2\workspace\Sportyshoes>docker run -d --name dockercontainer -p 80:80 dockerimage
99635203e312d729cb50de83655b550fa067170826ecb4158903a9875a44e446

C:\Users\amitya2\workspace\Sportyshoes>docker logs dockercontainer

:: Spring Boot ::
2023-03-06 09:31:23.046 INFO 1 --- [ main] com.demo.DemoApplication : Starting DemoApplication v0.0.1-SNAPSHOT using Java 1.8.0_342 on 99635203e3
v
```

←

→

↺

100.27.43.72:8080

☆


🔒

📄

🔴

📁

☰

 **Jenkins**

🔍 Search (CTRL+K) ⓘ

🛡️ 1

👤 admin ▾

🚪 log out

Dashboard >

+

New Item

👤 People

📅 Build History

⚙️ Manage Jenkins

📋 My Views

Build Queue ▾

No builds in the queue.

Build Executor Status ▾

1 Idle

2 Idle

All +

S	W	Name ↕	Last Success	Last Failure	Last Duration	
✓	☀️	<a href="#">DockerJenkinsProject05</a>	10 min #3	N/A	0.35 sec	▶️
✓	☁️	<a href="#">DockerJenkinsProject05Pipeline</a>	10 min #3	20 min #1	3 sec	▶️
✓	☀️	<a href="#">DockerJenkinsProject05PipelineScript</a>	9 min 52 sec #2	N/A	7.8 sec	▶️

Icon: S M L    Icon legend    📡 Atom feed for all    📡 Atom feed for failures    📡 Atom feed for just latest builds

Add description

←

→

↺

100.27.43.72:8080/job/DockerJenkinsProject05/

☆


🔒

📄

🔴

📁

☰

 **Jenkins**

🔍 Search (CTRL+K) ⓘ

🛡️ 1

👤 admin ▾

🚪 log out

Dashboard > DockerJenkinsProject05 >

📄 Status

</> Changes

📁 Workspace

▶️ Build Now

⚙️ Configure

🗑️ Delete Project

📄 Git Polling Log

✏️ Rename

🌞 Build History trend ▾

🔍 Filter builds... ⓘ

Project DockerJenkinsProject05

Add description

Disable Project

Permalinks

- [Last build \(#4\), 1 min 6 sec ago](#)
- [Last stable build \(#4\), 1 min 6 sec ago](#)
- [Last successful build \(#4\), 1 min 6 sec ago](#)
- [Last completed build \(#4\), 1 min 6 sec ago](#)

Jenkins

Search (CTRL+K)

Dashboard > Docker/JenkinsProject05Pipeline >

Status

Changes

Workspace

Build Now

Configure

Delete Project

Git Polling Log

Rename

Project DockerJenkinsProject05Pipeline

Add description

Disable Project

Permalinks

- Last build (#4), 1 min 22 sec ago
- Last stable build (#4), 1 min 22 sec ago
- Last successful build (#4), 1 min 22 sec ago
- Last failed build (#1), 30 min ago
- Last unsuccessful build (#1), 30 min ago
- Last completed build (#4), 1 min 22 sec ago

Build History

trend ▾

Filter builds... /

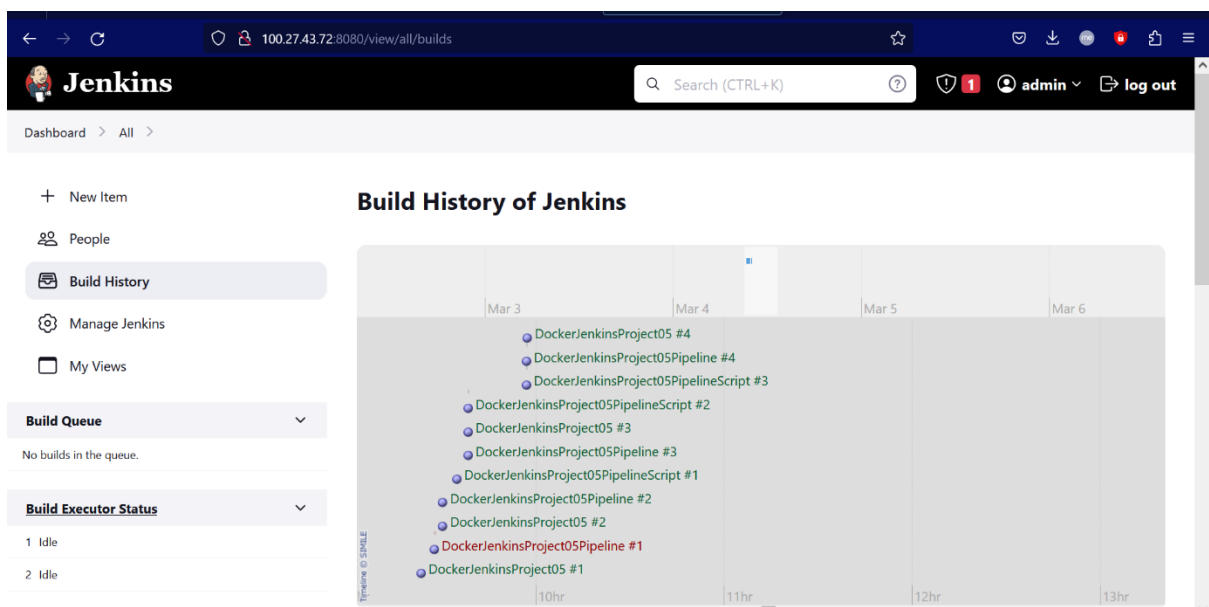
The screenshot shows the Jenkins web interface for a pipeline named "PipelineSyntax". The left sidebar contains navigation links for "Dashboard", "Build History", and "Pipeline Syntax". The main area displays the "Pipeline Syntax" view, which includes a summary of stage times and a table of recent builds.

	Source Code	Image	Application
<b>Average stage times:</b> (Average full run time: ~9s)	503ms	2s	4s
#3 Mar 04 15:26 No Changes	479ms	2s	5s
#2 Mar 04 15:08 1 commit	395ms	2s	4s
#1 Mar 04 15:04 No Changes	636ms	4s	3s

Below the table, there are three permalinks for the last build (#3):

- Last build (#3), 1 min 36 sec ago
- Last stable build (#3), 1 min 36 sec ago
- Last successful build (#3), 1 min 36 sec ago
- Last completed build (#3), 1 min 36 sec ago

Dashboard > All >				
2 Idle				
S	Build	Time Since	T	Status
✓	DockerJenkinsProject05 #4	2 min 40 sec		stable
✓	DockerJenkinsProject05Pipeline #4	2 min 42 sec		stable
✓	DockerJenkinsProject05PipelineScript #3	2 min 49 sec		stable
✓	DockerJenkinsProject05PipelineScript #2	21 min		stable
✓	DockerJenkinsProject05 #3	21 min		stable
✓	DockerJenkinsProject05Pipeline #3	21 min		stable
✓	DockerJenkinsProject05PipelineScript #1	25 min		stable
✓	DockerJenkinsProject05Pipeline #2	29 min		back to normal
✓	DockerJenkinsProject05 #2	29 min		stable
✗	DockerJenkinsProject05Pipeline #1	32 min		broken since this build



admin-controller Admin Controller		▼
GET	/admin/getallcustomerbydate/{date}	getAllCustomersByDateAndCategory
GET	/admin/getallcustomerbyname/{custname}	findByCustomerName
GET	/admin/getallproductsbycategory/{category}	getAllProductssByCategory
GET	/admin/getallusers	getAllUsers1
GET	/admin/getallvalidcustomers	getAllUsers
POST	/admin/insertadmin	InsertAdmin
PUT	/admin/updateadminbyid/{adminid}	updateAdmin

## customer-controller Customer Controller

**PUT** /customer/addcustomerproducts/{custid}/{prodid} updateProductInCustomer

**DELETE** /customer/deletcustomer/{userid} deleteCustomer

**GET** /customer/getallcustomers getAllCustomers

**GET** /customer/getcustomer/{userid} getCustomer

**POST** /customer/insertcustomer/{id} insertCustomer

**PUT** /customer/updatecustomerbyid/{userid} updateCustomer

## product-controller Product Controller

**DELETE** /product/deleteproduct/{productid} deleteProduct

**GET** /product/getallproducts getAllProducts

**GET** /product/getproduct/{productid} getProduct

**POST** /product/insertproduct insertProduct

**PUT** /product/updateproductbyid/{productid} updateProduct

## purchased-product-controller Purchased Product Controller

**DELETE** /purchased/deletepurchasedproduct/{productid} deletePurchasedProduct

**GET** /purchased/getallpurchasedproducts getAllPurchasedProducts

**GET** /purchased/getpurchasedproduct/{productid} getProduct

**POST** /purchased/insertpurchasedproduct insertProduct

**PUT** /purchased/updatepurchasedproductbyid/{productid} updatePurchasedProduct

## user-controller User Controller

**PUT** /user/addmoreuserproducts/{custid}/{prodid} updateProductInCustomer

**GET** /user/getproducts/{userid} getProducts

**POST** /user/insertproductinuser/{userid}/{prodid} insertProductUserInDB

**POST** /user/signup insertUser

**PUT** /user/updateuserbyid/{userid} updateUser



# SOURCE CODE

```
New Text Document - Notepad
File Edit Format View Help
> sudo chmod 777 /var/run/docker.sock

4. FOR INITIAL CREDENTIALS AND BASIC SETUP
-----
1. CLICK ON INSTALL SUGGESTED PLUGINS
2. PROVIDE YOUR CREDENTIALS
3. WELCOME TO JENKINS
4. MANAGE PLUGINS>INSTALL SUGGESTED PLUGINS>AVAILABLE PLUGINS>MAVEN INTEGRATION >CLICK ON INSTALL WITHOUT RESTART

5. PREPARE FREESTYLE PROJECT(CI-CONTINUOUS INTEGRATION)
-----
1. GOTO>DASHBOARD>create>new job>select free style project>give any name>click on ok
2. give description
3. source code manangement
   GIT:
   URL:https://github.com/Nikunj-Java/docker_master.git
   BRANCHES TO BUILD: */master or */main
4. Build trigger
   Poll SCM
   H/2 * * * *
5. click on apply and save

6. Build The project

6. AFTER JENKINS INSTALLATION
-----
1. Got to manage jenkins>>configure clouds and install docker and docker pipeline.
2. Create a freestyle project and go to advanced pipeline.
3. Integrate github and docker hub repository credentials.

7. DEPLOYMENT OF SPRING BOOT APPLICATION
-----
1. Goto public_ip_address of AWS EC2 instance and move to port 9090 to access its API CRUD Operations using Postman and Swagger-UI.

Ln 150, Col 1    100%    Windows (CRLF)    UTF-8

New Text Document - Notepad
File Edit Format View Help
STEP:6 JENKINS INSTALLATION
*****

-----

> curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
> echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list >
> sudo apt-get update
> sudo apt-get install jenkins

1. TO START WITH JENKINS
-----

> sudo service jenkins start
> sudo service jenkins status

CONNECT: GOTO>AWS>EC2>copy public ip address of your instance:8080 on browser and hit enter

2. TO GENERATE SECRET PASSWORD
-----
> sudo cat /var/lib/jenkins/secrets/initialAdminPassword

3. TO GIVE PERMISSION FOR ALL
-----

> sudo chmod 777 /var/run/docker.sock

Ln 150, Col 1    100%    Windows (CRLF)    UTF-8
```

```
New Text Document - Notepad
File Edit Format View Help
3. TO LIST THE DOCKER IMAGES
-----
> sudo docker images

4. To DEPLOY A DOCKER IMAGE
-----
> docker build -t <docker_image>

> docker run -p 9090:8082 <docker_image>

STEP:4 INSTALL JDK
*****
> sudo apt-get update
> sudo apt install default-jdk -y

to verify the installation

> java --version
*****
STEP:5 MAVEN INSTALLTION
*****
> sudo apt-get update
> sudo apt install maven -y

*****
STEP:6 JENKINS INSTALLATION
*****
-----
> curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

link:https://docs.docker.com/engine/install/ubuntu/
-----
> sudo apt-get update

> sudo apt-get install ca-certificates curl gnupg lsb-release

> sudo mkdir -m 0755 -p /etc/apt/keyrings
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

> echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |

> sudo apt-get update

> sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

1. TO VERIFY THE INSTALLTION
-----
> sudo docker -v
      OUTPUT: Docker version 23.0.1, build a5ee5b1
> sudo docker --version

> sudo docker info

2. TO LIST DOCKER CONTAINERS
-----
> sudo docker container ls

3. TO LIST THE DOCKER IMAGES
-----
> sudo docker images

4. To DEPLOY A DOCKER IMAGE
-----
```

```
New Text Document - Notepad
File Edit Format View Help
*****
STEP:1 AWS UBUNTU INSTANCE
*****
1. PREPARE AWS INSTANCE(UBUNTU SERVER 22.04 LTS (HVM) ,SSD VOLUME TYPE)
2. SECURITY : ADD PORT NO :80 WITH CUSTOM TCP IP
3. DOWNLOAD .PEM KEY TO CONNECT YOUR INSTANCE WITH YOUR LOCAL MACHINE USING MOBA X-TERM

*****
STEP:2 CONNECT USING MOBA X-TERM
*****
OPEN MOBA X-TERM
> cd d: //d: is my drive
> cd phase-5 //phase-5 is my folder where i have copied .pem key

> goto>aws>instance>choose your instance>connect>ssh>copy the example key>

>open moba x-term>right click> enter

*****
STEP:3 DOCKER INSTALLATION ON UBUNTU OS
*****
goto>Google>Docker Installation on Ubuntu OS

link:https://docs.docker.com/engine/install/ubuntu/
-----

> sudo apt-get update

> sudo apt-get install ca-certificates curl gnupg lsb-release

> sudo mkdir -m 0755 -p /etc/apt/keyrings
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

Ln 150, Col 1 100% Windows (CRLF) UTF-8 18:32 06-03-2023

01 Jenkins With Docker - Notepad
File Edit Format View Help

2. TO GENERATE SECRET PASSWORD
-----
> sudo cat /var/lib/jenkins/secrets/initialAdminPassword

3. TO GIVE PERMISSION FOR ALL
-----

> sudo chmod 777 /var/run/docker.sock

4. FOR INITIAL CREDENTIALS AND BASIC SETUP
-----

1. CLICK ON INSTALL SUGGESTED PLUGINS
2. PROVIDE YOUR CREDENTIALS
3. WELCOME TO JENKINS
4. MANAGE PLUGINS>INSTALL SUGGESTED PLUGINS>AVAILABLE PLUGINS>MAVEN INTEGRATION >CLICK ON INSTALL WITHOUT RESTART

5. PREPARE FREESTYLE PROJECT(CI-CONTINUOUS INTEGRATION)
-----
1. GOTO>DASHBOARD>create>new job>select free style project>give any name>click on ok
2. give description
3. source code manangement
GIT:
URL:https://github.com/Nikunj-Java/docker_master.git
BRANCHES TO BUILD: */master or */main
4. Build trigger
Poll SCM
H/2 * * * *
5. click on apply and save

6. Build The project

Ln 1, Col 1 100% Windows (CRLF) UTF-8 22:42 05-03-2023
```

```
01 Jenkins With Docker - Notepad
File Edit Format View Help
STEP:4 MAVEN INSTALLTION
*****

> sudo apt-get update
> sudo apt install maven -y

*****
STEP:5 JENKINS INSTALLATION
*****

> GOTO> GOOGLE> HOW TO INSTALL JENKINS IN UBUNTU
LINK: https://www.jenkins.io/doc/book/installing/linux/

-----

> curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
> echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list >
> sudo apt-get update
> sudo apt-get install jenkins

1. TO START WITH JENKINS
-----

> sudo service jenkins start
> sudo service jenkins status

CONNECT: GOTO>AWS>EC2>copy public ip address of your instance:8080 on browser and hit enter

Ln 1, Col 1 100% Windows (CRLF) UTF-8 22:42 05-03-2023

01 Jenkins With Docker - Notepad
File Edit Format View Help
STEP:2 INSTALL DOCKER
*****

> sudo apt-get update

> sudo apt-get install ca-certificates curl gnupg lsb-release

> sudo mkdir -m 0755 -p /etc/apt/keyrings
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

> echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
> sudo apt-get update

> sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

1. TO VERIFY THE INSTALLTION
-----

> sudo docker -v
    OUTPUT:Docker version 23.0.1, build a5ee5b1

*****
STEP:3 INSTALL JDK
*****

> sudo apt-get update
> sudo apt install default-jdk -y

to verify the installtion

> java --version

*****
STEP:4 MAVEN INSTALLTION
*****

Ln 1, Col 1 100% Windows (CRLF) UTF-8 22:42 05-03-2023
```

```
01 Jenkins With Docker - Notepad
File Edit Format View Help
*****
JENKINS WITH DOCKER
*****

STEP:1 CREATE AWS UBUNTU INSTANCE WITH PORT NO:8080

STEP:2 INSTALL DOCKER

STEP:3 INSTALL JDK

STEP:4 INSTALL MAVEN

STEP:5 INSTALL JENKINS

*****
STEP:1 CREATE AWS UBUNTU INSTANCE WITH PORT NO:8080
*****
> prepare AWS Ubuntu INSTANCE (Ubuntu Server 22.04 LTS (HVM) ,SSD Volume Type)
> Security : add port no :8080 with custom TCP rule
> Download.pem key to local machine and connect using moba x-term

> open moba x-term
> cd d:
> cd phase-5

> goto>aws>instance>your instance>connect>ssh>copy example key

> goto >mobax-term terminal>right click>hit enter

or connect directly from web browser

*****
STEP:2 INSTALL DOCKER
*****

02 Push Docker Image to DockerHub - Notepad
File Edit Format View Help
*****
STEP:1 LOGGED IN TO DOCKER HUB ACCOUNT
*****

create an Account on dockerHub:

Link: https://hub.docker.com/
-----

> sudo docker login

give username:
give password:
[Note: password is not visible]

if you are getting some permission denied error

> sudo chmod 666 /var/run/docker.sock

> sudo docker login

give username:
give password:
[Note: password is not visible]

*****
STEP:2 PUSH AN IMAGE TO DOCKER HUB
*****

EG: sudo docker tag <name of your image> <username>/<name of your directory-you can give any name>

> sudo docker tag dockerimage <docker_hub_name>/<image_name>
> sudo docker push <docker_hub_name>/<image_name>
```

```
> sudo docker pull mysql

*****
STEP:5 PULL GIT HUB IMAGES (CUSTOM IMAGES)
*****

LINK: https://github.com/Nikunj-Java/docker_master.git

> git clone https://github.com/Nikunj-Java/docker_master.git
> ls (to check available folders)
> cd docker_master

1. LET'S PREPARE THE IMAGE IN A DOCKER CONTAINER
-----
> sudo docker build -t dockerimage . (. is mandatory)
> sudo docker images (to check the image is prepared or not?)

2. LET'S RUN THE IMAGE IN A DOCKER CONTAINER
-----
> sudo docker run -d --name mycontainer -p 80:80 dockerimage

> sudo docker container ls

3. TO CHECK WITH APP IS RUNNING OR NOT?
-----
> curl localhost

this will open 'index.php' page of your app

goto>AWS>Instance>Copy Public Ip Address
goto>webbrowser>ipAddress:80
```

Ln 1, Col 1

100%

Windows (CRLF)

UTF-8

22:41

05-03-2023

```
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

> echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |

> sudo apt-get update

> sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

1. TO VERIFY THE INSTALLTION
-----

> sudo docker -v
      OUTPUT:Docker version 23.0.1, build a5ee5b1
> sudo docker --version

> sudo docker info

2. TO LIST DOCKER CONTAINERS
-----
> sudo docker container ls

3. TO LIST THE DOCKER IMAGES
-----
> sudo docker images

4. TO CHECK DOCKER VOLUME
-----
> sudo docker volume ls

*****
STEP:4 PULL DOCKER IMAGES
*****

> sudo docker pull ubuntu
```

Ln 1, Col 1

100%

Windows (CRLF)

UTF-8

22:41

05-03-2023

```
01 Docker Installation on EC2 Instance - Notepad
File Edit Format View Help
|*****
STEP:1 AWS UBUNTU INSTANCE
*****
1. PREPARE AWS INSTANCE(UBUNTU SERVER 22.04 LTS (HVM) ,SSD VOLUME TYPE)
2. SECURITY : ADD PORT NO :80 WITH CUSTOM TCP IP
3. DOWNLOAD .PEM KEY TO CONNECT YOUR INSTANCE WITH YOUR LOCAL MACHINE USING MOBA X-TERM

*****
STEP:2 CONNECT USING MOBA X-TERM
*****
OPEN MOB X-TERM
> cd d: //d: is my drive
> cd phase-5 //phase-5 is my folder where i have copied .pem key

> goto >aws>instance>choose your instance>connect>ssh>copy the example key>

>open moba x-term >right click> enter

*****
STEP:3 DOCKER INSTALLATION ON UBUNTU OS
*****
goto>Google>Docker Installation on Ubuntu OS

link:https://docs.docker.com/engine/install/ubuntu/
-----

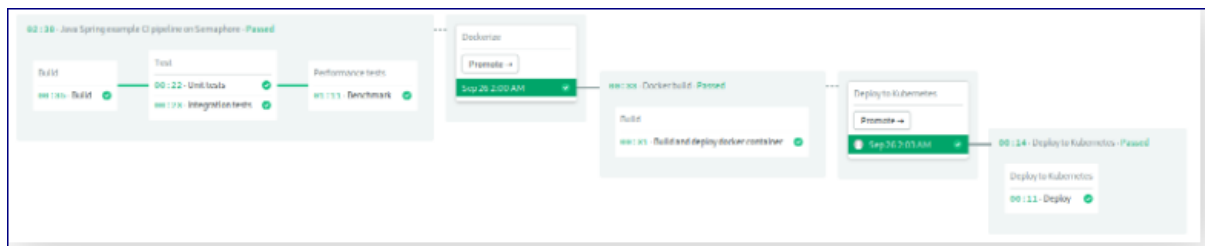
> sudo apt-get update

> sudo apt-get install ca-certificates curl gnupg lsb-release

> sudo mkdir -m 0755 -p /etc/apt/keyrings
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

Ln 1, Col 1 100% Windows (CRLF) UTF-8
22:41
05-03-2023
```

# Flowchart



TomFern : Docs Support

semaphore-demo-java-spring > Edit workflow (master)

Workflow Builder .semaphore/semaphore.yml .semaphore/docker-build.yml .semaphore/deploy-k8s.yml Dismiss and Exit Run the workflow

Deploy to Kubernetes

Dependencies

Only one block in the pipeline. You need at least two blocks to define dependencies.

Jobs

One command per line.

```
Deploy
cat deployment.yml | envsubst | tee deployment.yml
kubectl apply -f deployment.yml
```

+ Add another Job

Prologue

Executes before each job.

```
checkout
# add cloud specific install/auth commands here
# ...
```

TomFern : Docs Support

semaphore-demo-java-spring > Edit workflow (master)

Workflow Builder .semaphore/semaphore.yml .semaphore/docker-build.yml Dismiss and Exit Run the workflow

Java Spring example CI pipeline on Semaphore

Build

Test

Performance tests

Unit tests

Integration tests

Benchmark

Build

Dependencies

Test

Performance tests

Jobs

One command per line.

```
Build
checkout
cache restore
mvn -q package jmeter:configure -Dmaven.test.skip=true
cache store
```

+ Add another Job



## **Algorithm**

Step 1> Start.

Step 2> The user must create an instance of AWS.

Step 3> Once the instance is created and running, then perform installation of both Jenkins and Docker given in the source code for guide.

Step 4> After Installation, user need to clone the spring boot application from GitHub repository.

Step 5> Create the corresponding docker image for that spring boot application.

Step 6> After docker image has been created user can deploy his application using docker container in respective port number.

Step 7> Install the necessary plugins and clouds such as docker and docker pipeline for Jenkins safe deployment.

Step 8> Stop